CLAIMS

What is claimed is:

1. A method for computing partial differential equations in a hardware graphics pipeline, comprising:

receiving input in the hardware graphics pipeline; and

processing the input to generate a solution to the partial differential equation utilizing the hardware graphics pipeline.

2. The method as set forth in claim 1, wherein the input represents boundary conditions.

3. The method as set forth in claim 1, wherein the input includes textures.

4. The method as set forth in claim 1, wherein the input includes geometry.

5. The method as set forth in claim 4, wherein the geometry is selected from the group consisting of polygons, vertex data, points, and lines.

6. The method as set forth in claim 1, wherein the input includes a local area of textures.

7. The method as set forth in claim 6, wherein the local area of textures is generated by sampling a texture map.

8. The method as set forth in claim 6, wherein the local area of textures is filtered.

9. The method as set forth in claim 6, wherein the local area of textures is filtered utilizing a plurality of filters.

10. The method as set forth in claim 6, wherein the local area of textures is filtered utilizing a filter including a plurality of elements.

11. The method as set forth in claim 6, wherein the local area of textures is used to sample a texture map to generate a modified local area of textures.

12. The method as set forth in claim 1, wherein the processing includes a relaxation operation.

13. The method as set forth in claim 12, wherein the relaxation operation is selected based on the partial differential equation.

14. The method as set forth in claim 12, wherein the processing includes a plurality of iterations of the relaxation operation.

15. The method as set forth in claim 14, wherein a number of the iterations of the relaxation operation is reduced using at least one of a prolongation operation and a restriction operation.

16. The method as set forth in claim 12, wherein the processing further includes determining whether the solution has converged.

17. The method as set forth in claim 16, wherein it is determined whether the solution has converged after each iteration of the relaxation operation.

18. The method as set forth in claim 16, wherein it is determined whether the solution has converged after a predetermined number of multiple iterations of the relaxation operation.

19. The method as set forth in claim 16, wherein the determining whether the solution has converged includes calculating errors.

20. The method as set forth in claim 19, wherein the determining whether the solution has converged further includes summing the errors.

21. The method as set forth in claim 19, wherein the determining whether the solution has converged further includes concluding that the solution has converged if the error is less than a predetermined amount.

22 The method as set forth in claim 16, wherein if it is determined that the solution has converged, repeating the processing using an altered parameter value.

23. The method as set forth in claim 14, wherein the number of iterations of the relaxation operation is determined prior to the processing.

24. The method as set forth in claim 8, wherein the filtering is carried out using a programmable filter.

25. The method as set forth in claim 8, wherein the filtering is carried out using a non-programmable filter.

26. A system comprising a hardware graphics pipeline for processing input to generate a solution to partial differential equations.

27. A method for computing partial differential equations in a hardware graphics pipeline, comprising:

means for receiving input in a hardware graphics pipeline; and

means for processing the input to generate a solution to a partial differential equation utilizing the hardware graphics pipeline.

28. A method for computing partial differential equations in a hardware graphics pipeline, comprising:

receiving boundary conditions;

computing a solution to the partial differential equations utilizing a relaxation operation involving the boundary conditions, at least some the computing done in the hardware graphics pipeline;

determining whether the solution has converged; and

if the solution has not converged, repeating the computing and the determining.

29. A method for computing partial differential equations in a hardware graphics pipeline, comprising:

receiving boundary conditions in the form of at least one of geometry and textures;

computing a solution to the partial differential equation utilizing a relaxation operation involving the boundary conditions, at least some the computing done in the hardware graphics pipeline;

determining whether the solution has converged by:

calculating errors,

summing the errors, and

concluding that the solution has converged if the sum of errors is less than a predetermined amount;

if the solution has not converged, repeating the computing and determining;

if the solution has converged, incrementing a time value; and

repeating the foregoing operations using the incremented time value.

30. A method for generating a 3-D graphics image, comprising:

receiving a first input into a hardware graphics pipeline;

processing the first input to generate a solution to a partial differential equation utilizing the hardware graphics pipeline;

receiving a second input into the hardware graphics pipeline;

rendering the 3D graphics image utilizing the hardware graphics pipeline, wherein the rendering utilizes the second input and the result of the processing of the first input.

31. The method as set forth in claim 30, wherein:

the first input comprises boundary conditions; and

the processing comprises:

computing a solution to the partial differential equation utilizing a relaxation operation involving the boundary conditions;

determining whether the solution has converged; and

if the solution has not converged, repeating the computing and determining.